The Mathematics Education into the 21st Century Project
Proceedings of the International Conference
The Decidable and the Undecidable in Mathematics Education
Brno, Czech Republic, September 2003

Towards convergence of mathematics and computer science education

Alexander Khait
Jerusalem College of Engineering, Ramat Beit HaKerem, POB 3566, Jerusalem 91035, Israel
e-mail: khait@mail.jce.ac.il

ABSTRACT. Mathematics and computer science are strongly intertwined, and each subject could be seen as complementary to the other. Most pupils in mathematics classes on the secondary level expect to use it in computer-related applications. The paper explores possibilities of teaching these two subjects in a uniform fashion, so that each subject helps a better understanding of the other.
1. A computer metaphor replaces the classical "platonic" nature of mathematical objects, as they are presented to students, by an ostensive model. This approach stresses algorithmic nature of mathematics on the one hand, and facilitates introduction of the basic concepts of computer science, on the other hand.
2. The language of the set theory with its relatively simple syntax and independence on technical features of specific programming languages provides a suitable basis for introducing students to formal language needed for work in computer-related industries with its incessant transitions between formal (human-computer) and informal (human-human) communications.
I conclude with a description of my work with mathematics teachers trying to advance the above ideas.

I. Introduction

As a result of the computer revolution there is a large increase in high school graduates that continue their professional education in mathematics-related subjects. The needs of these people are different from those who never study mathematics after secondary school, as well as from professional mathematicians. From the point of view of a mathematics teacher there has never been so large a population that is not naturally inclined to study mathematics but needs it professionally. So the challenge to the science of mathematics education posed by the modern society in this context can be personalized in a student that is not naturally inclined to mathematics but has to receive solid mathematical background to be prepared to work in computer-related technologies.
Two chief aims of secondary school mathematics (except applications) are teaching students to work with a symbolic language (algebra) and the mathematical way of thinking associated with exact formulations and logical deductions. Since there are almost no theorems in the school-level algebra, this purpose has been traditionally fulfilled by geometry. Lately geometry began to be seen as a tool to teach students about space rather than about formal thinking. The reason is that translation of geometry to a rigorous axiomatic theory (accomplished by Hilbert) is unsuitable for presentation to non-specialists.
One can discern two points where computer revolution requires changes in emphasizes in school mathematics education: 1) stress on algorithmic nature of mathematics; 2) competence in formal communications. Both a symbolic language and the mathematical way of thinking are crucial for these goals. However, mathematics needed for computer specialists is very different from traditional mathematics. “The sort of mathematics that arises in a computing context is not necessarily what most people would consider to be mathematics at all. Its character may seem like that of ‘mere’ organization, symbol management, or data manipulation” Truss (1999, Preface). So major changes in emphasizes should be made in each of the two directions. Some of these changes, in particular those in teaching algebra, can help to present mathematical subjects in clearer, less abstract terms. Let us discuss these changes.
One of the main difficulties in understanding mathematical discourse involves dealing with "platonic" objects with no tangible correlates. The traditional way of teaching mathematics is based on abstract thinking in terms of these objects. Abstractness of mathematics (and in particular that of algebra) is a perpetual root of negative feelings towards mathematics among generations of high school students. This abstractness stems from the platonic model of mathematics, which *is* natural to mathematics teachers. "A basic and often implicit underlying philosophy of mathematics in teaching is that of 'sufficiently liberal platonism' which teachers have acquired during their university studies and then taken with them to school." (Seeger and Steinbring, 1994). My experience is that many students who would like to continue their studies and work in computer-related fields believe that "**computers are a part of our life while mathematics is not**!" I think that platonism plays an important part in creating this feeling of mathematics irrelevance.
Recent developments in mathematics education show that reification, i.e. thinking in terms of objects, is an important part in absorption of new material (Sfard, 1991, Dubinsky, 1995). The nature of mathematical objects is a serious philosophical problem. Most working mathematicians (and certainly

the vast majority of mathematics teachers) are not interested in this question, may be because there is no solution in view. It should be noted that there is no agreement even on the question what is a natural number (see classical work by Benacerraf, 1965, and Steinhart, 2002, for latest developments). It is only natural that our students have difficulties in creating models of unknowns, parameters, etc. So it is worthwhile to supply them ostensive models of mathematical objects. "One of the many features of computer programs … is that they present objects. Thus they can be used to support reification" (Mason, 1989). Moreover, a computer itself is a tangible object, so a model of mathematics based on a computer metaphor presents an opportunity to replace the classical platonic model in relevant areas of mathematics education. In sec.II I bring an example of algebra presented in terms of a computer metaphor.

The most important component of the mathematical way of thinking for workers in high-tech industries is the ability to express themselves in a rigorous language, where each word and expression has an unambiguous meaning. Professional communications with a computer (defining problems, formulating models, writing computer programs) proceed in such language. This need implies serious changes in the relative importance of various parts of mathematical discourse as it appears in the classroom (Khait, 2003):

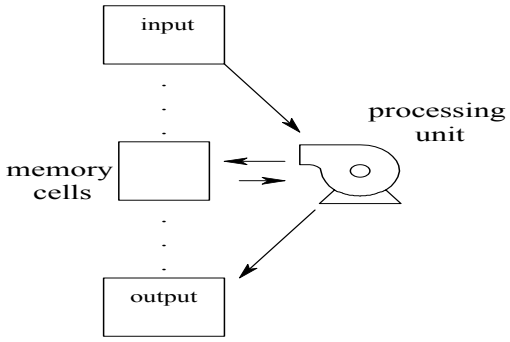
a) The classical use of definitions in mathematics is in introducing new concepts (Vinner,1991). The computer revolution has introduced a new aspect to the role of definitions. A competent professional has to construct new definitions of various computer objects (e.g. structures or classes) and understand definitions constructed by colleagues. Definitions in computer programming have different levels of importance and depth: one does not expect to develop a deep intuitive understanding of each object created for particular needs of an occasionally encountered program; one has, however, to understand these definitions, that is, their meaning in terms of operational consequences. Thus, the work with definitions has to become a purpose in itself rather than a more or less appropriate means for something else.

b) Programmers usually do not prove the correctness of their algorithms. However, to become a good programmer one has to develop intuition and logical thinking to analyze programs. Working on proofs in an appropriate context helps to develop such intuition. So the main reason for teaching proofs to this population is that a proof helps students to assure themselves that the statement under consideration is indeed true and to develop relevant intuition. Statements relevant for work in computer environment are very different from geometrical propositions: there is only a play with abstract definitions with no visual component.

In sec.III I present examples of using the set-theoretical language to develop skills of formal communications and logical deductions.

II. Computer as an ostensible metaphor in mathematics education

Here is a simplified, "virtual" computer that can help to reify algebra. It could be seen as a device somewhat like a food processor: basic frame with modular hardware. The hardware is composed of cells (called in computer jargon memory cells) and processing units. Inputs are inserted in special input cells. The content of these cells can be copied for



further processing to other cells whose content can be both copied and modified. An output is returned at the outlet (e.g. special output cells) and can be taken (copied) from there for any use. For each type of object there are cells of a suitable structure. More complex cells are built to contain more complex objects. These cells are formed by arrangement of simpler cells. For each processing unit there is a specific type of allowed input, so that only objects of this type can serve as inputs. No two units can be used simultaneously. New units can be added to a device.

Computer model "invites" writing mathematics in algorithmic style that helps to emphasize its algorithmic nature. Teaching algebra along these lines transfers the stress from numerical problems to solutions of generic (parameterized) problems. A natural way to do mathematics in algorithmic style is writing in simple pseudocode. Such pseudocode, used below, is based on the four standard control structures (assignment, conditioning, FOR-loop and WHILE-loop) (Harel, 1993).

Example. Linear equations.

We begin with a simple linear equation and later its solution algorithm will be used to solve a more complex linear equation. equation ax=b

Regular style	Algorithmic style
1. a=0	**SimpleLinearEquation**(a,b) [*SLE*(a,b)]
1_a b=0: x any real number	IF a=0 THEN IF b=0 THEN RETURN (-∞,∞);
1_b b≠0: No Solution	ELSE RETURN "No Solution";
2. a≠0: x=b/a	ELSE RETURN b/a;

Apparently the two representations are rather similar, however, there are important differences in meanings assigned to various symbols helping to create tangible models for mathematical objects:

1. Parameters *a* and *b* in *SimpleLinearEquation*(a,b) [*SLE*] are inputs inserted in specific cells.
2. The computed value of the unknown is the output. The command 'RETURN b/a' leads to a delivery of the result at the outlet. Thus, a clear distinction between parameters (as inputs) and unknowns (as outputs) is established.
3. *SLE* is an algorithm object (see citation from Mason above).
4. The functional form of writing *SLE*(a,b) emphasizes the dependence of the solution on parameters.
5. Algorithmic style naturally focuses attention on special cases encouraging use of the "IF" clause, while in the traditional way the unconditional "x=b/a" answer is a common mistake.

Furthermore, this algorithm can be used in solution of other equations, turning attention to reduction as an essential tool of mathematics: the "return to the previous problem" line does not usually appear in standard mathematics textbooks, while using previously written procedure is natural in algorithmic style:

equation $a_1x+b_1=a_2x+b_2$

Regular style	Algorithmic style
$a_1x+b_1=a_2x+b_2$	**LinearEquation**(a_1,b_1,a_2,b_2)
$(a_1-a_2)x=b_2-b_1$	a:=a_1-a_2; b:=b_2-b_1;
return to the previous problem	Sol:=*SLE*(a,b);
	RETURN Sol;

Let me summarize. The work on expressions with parameters reveals the algorithmic nature of algebra. A computer style representation of solution algorithms both emphasizes the algorithmic features of algebra and highlights hierarchical nature of mathematics (i.e. a reduction of more complicated problems to simpler ones). After seeing initial examples students should be encouraged to write new algorithms themselves. This leads to a shift of the teaching process from solutions of equations with numeric coefficients towards designing algorithms for equations with parameters. From the computer metaphor point of view a solution of an individual numerical equation amounts to inserting specific inputs and switching on a suitable processing unit. In practical terms it means to perform a manual run of the corresponding algorithm. Manual execution of explicitly written algorithms is in itself an important skill, essential for work in computer environment. Using a simple model of a "virtual computer" we can explain the meaning of unknowns, parameters and solutions of equations and inequalities in terms of ostensible objects. A pseudocode, in contrast with a real programming language, does not shift the attention from mathematics to interaction with a computer so that mathematics remains mental activity independent of actual technical devices.

III. Preparing students to communicate with computers

Let us discuss separately work with definitions and theorems.

Definitions.

Out of the various parts of mathematical discourse, the most radical changes should happen in a relative importance of formal definitions. This is because defining becomes a standard working tool used to introduce occasional definitions needed for some specific purposes. In terms of concepts and concept images (Tall and Vinner, 1981) mastering mathematical language moves the stress towards atomic concept images, associated with standard logical connections and quantifiers (Khait, 2003), which serve as bricks for definition construction. To illustrate what I mean by learning to work with "occasional" definitions here is an example of a question from the final examination of the course "Introduction to mathematical thinking" that uses the elementary set theory and the theory of relations as its subject matter. The course is given during the first semester to the first year college students (software engineers and industrial management engineers) and to mathematics teachers.

Consider the following definition

Let A_1,A_2 be disjoint sets, $A=A_1\cup A_2$. We say that a relation R defined on A has property # if every $a\in A$ satisfies the following conditions

1) for every $b\in A_2$ $(a,b)\in R$ AND $(b,a)\notin R$;
2) for every $b\in A_1$ $(a,b)\notin R$ AND $(b,a)\in R$.

a. Define the property of NOT #.

b. Give an example of a relation with the property # on the set A={1,2,3,4}.

Proofs.

While in the discussion of definitions the formal aspect was stressed, proofs are important mainly for human use and so here the intuition should be emphasized. The elementary set theory provides a lot of simple statements that could be proved or refuted by writing a line or two of mathematical text. During a single lesson the student can work on ten theorems/counterexamples. These proofs and refutations emphasize basic logical structures used in mathematics and in formal communications in general.

Here are examples of such statements: Let R and S be two relations over a set A. Prove or refute by an example the following statements

1. If R and S are transitive, then a. $R \cup S$ is transitive; b. $R \cap S$ is transitive.
2. If $R \cup S$ is transitive, then R and S are transitive.
3. If $R \cap S$ is transitive, then R and S are transitive.

Now one can substitute "transitive" by "reflexive" or "symmetric" or "antisymmetric" and so obtain another bunch of true or false (but always simple) theorems to help students to learn proper mathematical thinking.

Let me summarize. Capabilities to communicate in a formal language should be in focus of mathematics education of the students under consideration. A typical situation when a mathematical idea expressed in the classroom "is seldom the pupil's expression of the pupil's formulation of the pupil's idea" (Pimm, 1987) cannot be solved by reduction of rigor, because vague formulations are useless in computerized environment. Informal additional explanations in the form of "hand waving" cannot help either in communication with an inhuman object or with a colleague not present in the room. In many cases there is no other form to express a non-trivial technical idea but that of the standard mathematical language. And if such a form is unavailable, "the bodiless thought will return to the palace of shadows"(Vygotsky, 1934).

IV. Conclusions

The purpose of my work is to unify teaching of mathematics and computer science. I think that the move should be made by mathematics education since *it* has to make itself relevant to students and since the following relation basically holds:

((Computer Science) – (Technical Details)) $\subseteq$Mathematics

My experience of work with secondary school mathematics teachers, computer science teachers and with first year students in engineering colleges is of increasing divergence between the needs of the students and approaches of the teachers. Mathematics teachers who are unaware of computer-related applications of mathematics create an impression of irrelevance of their subject among students interested to work in high-tech industries. Many mathematics teachers do not like computers and even put it as a reason for becoming mathematics teachers. ("Otherwise I would become a programmer with a better pay and status.") Teachers deterred from computers naturally discourage student development in this direction. On the other hand, computer science teachers without proper mathematical background often see their job as focused on technicalities of particular languages and software neglecting deeper nature of their subject.

At first encounter with algorithmic style many of my teacher students react negatively. However, after the basic rules of pseudocode writing are understood and advantages of this approach are exposed, there is usually a clear change of mind of a rather large part of the class, which is not bad taking into account the well-known conservatism of experienced teachers. Some teachers become enthusiastic implementing the algorithmic style, in particular because it helps to establish better contacts with their pupils who learn programming. The set-theoretical language is not foreign to secondary school mathematics teachers, however, many of them lack practical experience with those logical structures that are not usual in standard geometric proofs.

References

Benacerraf, P. (1965), What numbers are not, Philosophical review **74**(1),47-73.

Dubinsky, E. (1991), Reflective abstraction in advanced mathematical thinking (in Advanced mathematical thinking, D.Tall (ed), Kluwer Academic publishers), 95-125.

Harel, D. (1993), Algorithmics, the spirit of computing, Addison-Wesley.

Khait, A. (2003), Goal orientation in mathematics education (to be published in *International Journal for Mathematics Education in Science and Technology*)

Mason, J. (1989), Mathematical abstraction as the result of a delicate shift of attention, *For the learning of mathematics* **9**(2),2-8.

Pimm, D. (1987), Speaking mathematically. Routledge&Kegan Paul, London.

Seeger, F. and Steinbring, H. (1994), The myth of mathematics, in P.Ernest (ed) Constructing mathematical knowledge, Falmer Press, London, pp 151-169.

Sfard, A. (1991), On dual nature of mathematical conceptions: reflections on processes and objects as different sides of the same coin, *Educational Studies in Mathematics*, **22**, 1-36.

Steinhart, E. (2002), Why numbers are sets, *Synthese* **133**,343–361.

Tall, D. and Vinner, S. (1981), Concept image and concept definition with particular references to limits and continuity, *Educational Studies in Mathematics*, **12**: 151-169.

Truss, J. (1999), Discrete mathematics for computer scientists, 2nd ed, Addison-Wesley.

Vinner, S. (1991), 'The role of definitions in teaching and learning mathematics' (in Advanced mathematical thinking, D.Tall (ed), Kluwer Academic publishers), pp.65-81.

Vygotsky, L.S. (1934), Language and Thought, Gosizdat, Moscow (translation from Psycholinguistics, Holt, Rinehart and Winston, New York, 1961, p.509)